

Modélisation et interrogation de données de capteurs à localisation mobile

Iulian Sandu Popa, Karine Zeitouni

PRISM Laboratory, University of Versailles
45, avenue des Etats-Unis - 78035 Versailles, France
{Iulian.Sandu-Popa, Karine.Zeitouni}@prism.uvsq.fr

Résumé. Les bases de données des objets mobiles constituent un important thème de recherche ces dernières années. De nombreux travaux ont porté sur la modélisation, le langage de requêtes ou l'indexation. Cependant, un objet mobile – tel qu'un véhicule – peut être muni de capteurs permettant de générer différentes mesures relatives à son état ou à son environnement tout au long de son déplacement. Gérer ces données est d'une importance majeure pour de nombreuses applications, telles que l'analyse du comportement des conducteurs en sécurité routière ou la reconstruction des circonstances d'accidents en sécurité routière, ou l'identification, par le biais de capteurs de vibrations, de défauts le long des rails en maintenance ferroviaire. Or, cette gestion n'est pas supportée par les approches existantes. Dans cet article, nous proposons un nouveau modèle de données et un langage pour gérer et analyser les données issues de capteurs géolocalisés mobiles. Pour y parvenir, nous introduisons un concept de profil qui capture la variabilité spatiale d'une mesure, de même que des opérations spécifiques permettant d'analyser les données. Nous décrivons également son implémentation au sein d'un SGBD objet-relationnel.

Abstract. Moving objects databases are an important research topic in recent years. A lot of work dealt with modeling, querying and indexing objects that move freely or in networks. However, a moving object – such as a vehicle - could report some measures related to its state or to its environment, which are sensed throughout his movement. Managing such data is of major interest for some applications such as analyzing driving behavior or reconstructing the circumstances of an accident in road safety, or identifying, by means of a vibration sensor, the defects along a railway in maintenance. However, this management is not covered by the existing approaches. In this paper, we propose a new data model and a language that handle mobile location sensor data. At this end, we introduce the concept of measure profile to capture the measure variability in space, along with specific operations that permit to analyze the data. We also describe their implementation using object-relational paradigm.

Keywords: Spatiotemporal databases, modeling, moving objects, query language, sensor data flows.

1. Introduction

Integrating mobile technology and positioning devices (GPS, cell phone) has led to produce a large amount of moving object data every day. A wide range of applications like public transportation, traffic management, location-based services, relies on these data. Besides, a moving object can easily be equipped with sensor devices that report on its state or on its environment. The generated mobile location sensor data are meaningful for many applications such as reconstructing the circumstances of an accident in road safety, identifying defects from vibration sensors along a railway in maintenance, or analyzing the exposure to hazard (e.g. pollutant) along a trip. As an example, in the field of road safety, the observation of natural driving behavior (on normal route for usual journeys) tends to replace the tests on simulators or those limited to dedicated circuits. Known as "naturalistic driving", these studies are based on data collected on a large scale and over a significant period of time [1].

However, studies reported in the literature have limited the volume of data and the possibilities of their exploitation. Thus, in a naturalistic driving campaign carried out on 100 cars, undertaken by the administration of U.S. Highway Safety [11], the storage is limited to situations of near-accident. As emphasized in their report [11], a large-scale database would be very useful to researchers and engineers to study the driving behavior and contribute to improving the vehicle equipment and road planning.

However, the challenge of a large-scale study is the management and analysis of a large mass of spatio-temporal data. A database system that supports this type of data and efficient querying is needed. We aim to study and develop such a database management system. This subject closely joins the field of databases for moving objects. However, the moving object - here the vehicle - is associated with additional data describing the state of the object or measures (speed, acceleration, steering wheel angle, the action on the breaking or gas pedals, etc.) recorded throughout his trip. These measures are variable in space and time. For the type of applications we address, the measures are more important than the mere spatio-temporal location. However, all work on mobile object databases consider only the location of the moving object and can not be generalized to measures ranging along a spatio-temporal trajectory. Moreover, although these values initially correspond to a temporal data stream, their variation is more dependent on their location on the network than time: for example, the variation of speed is usually constrained by the geometry of the road and the speed limit. Also, the temporal analysis of data of different trajectories is irrelevant because on the one hand they are asynchronous and on the other hand, this comparison makes sense only if these paths overlap in space. Therefore, we must capture the spatial variability of these measures and allow its manipulation through the data model and query language.

To our knowledge there is no such proposal in the state of the art. Nevertheless, among the works on moving objects databases (MOD), the one proposed in [6] provides a solid basis for modeling and querying moving objects. The idea of representing in a continuous way the temporal variation of the location or scalar values permits a good abstraction of moving objects. We extend this approach to capture the continuous spatial variation of scalars. The extension of the existing algebra consists in a new type and several classes of operations. The type captures the variation in space of any measure, which includes data from mobile sensors as a particular case.

This article provides the following contributions. First, we present a new concept of "space variant measure" and show its usefulness in the context of a naturalistic driving study. Second, we create a data model as an extension of the one proposed by Güting et al. [5]. Finally, we extend the existing query language with new classes of operations that are necessary in this novel application context. Besides the aforementioned application, the proposed model is interesting for other applications that generate and/or operate geo-localized

data streams. This is the case of rail (the TGV in France has many sensors), air or sea transportation. The measures can be observed or calculated and may be related to a trajectory of an object or a location. Thus, we can reason about the speed of a moving object, on the legal speed or inclination of a road or on the adherence at each location of the road depending on the weather. We can model the fine data on the mobility (where, when and at what speed) of vehicles (equipped with GPS), freight (labeled by RFID), or persons (GSM / GPRS) and meet the needs of management applications for fleets or road traffic, logistics and design of mobile networks.

The rest of this paper is organized as follows: we summarize the related work in section 2. Section 3 describes the proposed model. It presents the new type and the corresponding operations, and demonstrates its use by expressing some query examples. Section 4 describes our implementation with an emphasis on the aggregate operations. Finally, Section 5 concludes and offers directions for future work.

2. Related Work

The management of MOD has received particular attention in the recent years because of the advances and the omnipresence of mobile and geo-location technologies, such as cellular phones or GPS. Many works focus on modeling and language. We mention the work of Yeh's thesis [23], those relating to the project Chorochronos [4], [6], [10] and the work of the Wolfson's team [20], [21], [22]. Güting's book is a summary of progresses in this area [7] and [16] summarizes the spatio-temporal models including the moving objects model.

The target applications impact the model and the language in these proposals. We distinguish two types of applications. The LBS (Location Based Services) application type requires continuous or predictive queries, based on the current positions of moving objects. The pioneers are [20] whose model MOST (Moving Objects Spatio-Temporal) describes databases with dynamic attributes that vary continuously over time. They also propose a language called FTL (Future Temporal Logic) based on temporal logic to formulate predictive queries. The second type of applications concerns the analysis of complete spatio-temporal trajectories, using queries combining temporal and spatial intervals. The work of Güting [6] is an important reference point today. Various implementations exist, as in SECONDO [3] and STAU [13], then in HERMES-MDC [14], [15]. STAU is the first implementation to be based on object-relational database extensibility by providing a spatio-temporal data cartridge for Oracle [12]. However, these studies do not take into account the constrained movements of moving objects, such as the trajectories of vehicles in a road network or trains on a rail network. This aspect is essential for many applications, including those considered here.

Indeed, given a network, a constrained trajectory can be represented by the relative positions on the lines (e.g. roads) of this network. Again, the most comprehensive proposal is the one in [5]. Although the non-constrained (two-dimensional) model can be applied to the constrained trajectories, this is unwise for several reasons. The first is that the 2D model does not capture the relationship between the trajectory and the network space, while this information is essential for analysis. The second is that it limits the representation of the trajectory, estimated by linear interpolation between the reported positions, while the MO follows in fact the geometry of the network. In addition, the constrained model allows for dimensionality reduction by transforming the network in a 1D space by juxtaposing of all line segments [17]. This leads to better storage and query performance than with the free trajectory model. Finally, in the constrained model the trajectories can be easily described with a symbolic representation as a sequence of traversed lines and time intervals, which is less detailed but more intelligible and more compact [19].

2.1. Moving Objects Model

In this article, we are concerned by historical data of moving objects that are constrained by the network. The most comprehensive proposal to model the historical MO is, in our view, the framework of Güting [5]. Indeed, this proposal covers the abstract modeling, language and implementation. Moreover, it explicitly models the constrained moving objects and the relative position on the network. As discussed below, our proposal is based on this model and extends it with specific data for mobile sensors. Therefore, we summarize this model in the rest of this section.

Güting et al. propose an algebra defined by a set of specific types (Table 1) and a collection of operations on these types [5], [6]. The types are: scalar types (*BASE*), 2D space types (*SPATIAL*), network space related types (*GRAPH*), scalar or spatial types varying in time (*TEMPORAL*). Examples of types are: *real*, *point* (2D position), *gpoint* (position on the network), *gline* (line on the network), *moving(point)* (2D position varying in time) and *moving(gpoint)* (network position varying in time). All base types have the usual interpretation. For example, if we note with A_α the carrier set (definition domain) for the type α , then for the *real* type the carrier set is: $A_{real} = R \cup \{\perp\}$, where $\{\perp\}$ is null (or undefined). The time is isomorphic to the real numbers. The *range* data types are disjoint intervals and are used to make projections or selections on the *moving* types. Spatial types describe entities in the Euclidean space, while for the *GRAPH* types the space is represented by a network space. 2D types mainly correspond to standard definitions [8].

Table 1. The types defined in [5], [6].

\rightarrow <i>BASE</i>	<i>int, real, string, bool</i>
\rightarrow <i>SPATIAL</i>	<i>point, points, line, region</i>
\rightarrow <i>GRAPH</i>	<i>gpoint, gline</i>
\rightarrow <i>TIME</i>	<i>instant</i>
$BASE \cup SPATIAL \cup GRAPH \rightarrow$ <i>TEMPORAL</i>	<i>moving, intime</i>
$BASE \cup TIME \rightarrow$ <i>RANGE</i>	<i>range</i>

GRAPH types depend on the underlying network. Basically, the proposed model defines a network as a set of routes and junctions between these routes. A location in the network is a relative position on a route. It is described by the identifier of the route, a real number giving the relative position and one side. This is directly related to the concept of linear referencing widely used in transportation applications and implemented in systems such as Oracle [12]. The types *gpoint* and *gline* are represented in this manner. Finally, from the *BASE*, *SPATIAL* and *GRAPH* types, they derive the corresponding temporal types, using the type constructor *moving*. The temporal types are functions or infinite sets of pairs (instant, value). Such an infinite representation conceivable in the abstract model can not be implemented directly. In [3], a discrete representation is proposed for these types. We will discuss this aspect in more detail in section 4 of the paper.

A collection of operations is defined on the above data types. To avoid the proliferation of operations, one operator applies to several types. A set of non-temporal operations is first defined. Then, a process called *lifting* allows generating the corresponding temporal operations. Thus, all operations on non-temporal types are extended to the temporal ones. Finally, specific operations are added to manage the temporal types. In the context of constrained network trajectories [5], some new operations, such as **distance**, have been

adapted for *gpoint* and *gline* types (distance by route). New classes of operations are also added to analyze the interaction between the network and the 2D space, as well as specific operations such as computing shortest paths in the network. One can refer to [5], [6] for more detail.

However, to our knowledge, there is no work that addresses the general problem of managing mobile objects equipped with sensors, such as the considered type of application.

3. Proposed Model

3.1. Motivation and Examples

In this section, we present a real application that has motivated our work. A query scenario is used as an example throughout this article.

As indicated in the introduction, naturalistic driving studies have become more popular within the last years. These studies are essentially based on data gathered in normal (natural) driving conditions. They become economically possible thanks to the existing equipment in modern vehicles. Indeed, the large number of in-vehicle sensors is accessible via an interface (CAN bus) on which it is possible to connect an in-vehicle data logger. The CAN bus provides access to several measures including speed, acceleration, steering wheel angle, the action on the breaking or gas pedals, etc. The recording device can also receive data streams from other sources, such as a GPS sensor or radar (giving the distance to adjacent vehicles). This provides a comprehensive database on natural driving on the road. This database can provide valuable information on the use and utility of driver assistance systems (ABS, EPS, etc.) and can highlight near-accident (near-crash) situations. Moreover, according to the principle of black boxes on airplanes, it will provide information prior to an accident.

INRETS (French acronym for “National Institute for Research on Transport and Safety”) has developed a data logger (DIRCO) for naturalistic driving campaigns [2]. This is an on-board recording device connected to the vehicle’s CAN bus. It records measures such as: vehicle speed, speed of each wheel, longitudinal acceleration, odometer, steering wheel angle, brake pedal (0/1), ABS (0/1), etc. DIRCO offers the possibility of connecting other data sources as well, such as a GPS, an inertial station measuring the 3D acceleration and angle of the vehicle, or cameras¹. DIRCO acquires each data stream as a time sequence. The data from a source are stored in a specific file and each record is a tuple: $(t_i, \alpha_i^1, \alpha_i^2, \dots, \alpha_i^n)$, where t_i is the i^{th} time instant and α_i^k is the i^{th} value provided by the k^{th} sensor. As a detail, DIRCO allows an acquisition at very high frequency of up to 10ms cycles. The data flows from different sources are asynchronous.

While it may function as a black box for vehicles in order to reconstruct the circumstances of an accident, the DIRCO is primarily a research tool that can help to analyze the driving behavior, vehicle safety and diagnose problems related to road infrastructure. Its 16GB of flash memory allows data acquisition, camera off, for several months. A simple scenario is to equip several vehicles such as buses or cars with DIRCO, retrieve and centralize this data and then analyze it in order to identify behavioral patterns of driving.

This type of approach is also appropriate to the evaluation of recently emerged ADAS (Advanced Driver Assistance Systems). Whether the system is already well known as a GPS or speed control device, or it is an experimental system such as obstacle detection, all require an accurate and extensive assessment of their impact on driving. The European Commission

¹ The video stream is stored separately and its management is not in the scope of this paper

is funding since 2008 large-scale projects to evaluate mature technologies in the category of "intelligent transportation" systems. A particular aspect of these projects is to recourse to systematic collection of driving data with devices similar to DIRCO. Among these projects there is euroFOT (<http://www.eurofot-ip.eu/>) where hundreds of topics will be studied for at least a year. The data will be analyzed to assess the studied systems and draw conclusions regarding their generalization to future generations of vehicles.

Given this kind of application, one can easily understand the importance of developing a database adapted to the characteristics of these data, whether in relation to the storage volume, or in relation to geo-localized and temporal features. The different types of studied systems induce a large variability in the methods of analysis and often involve a high level of required detail (e.g., situations of near-accident). Some indicators can be calculated by using common database management systems, but sometimes at the cost of heavy programming and prohibitive computational time. In addition, no system seems at present able to manage speed profiles² (or any other information) measured at different times and positions but on the same road. However, a large number of queries in this context need this kind of approach.

To illustrate the contribution of our model in this context, we refer throughout the article to the following typical queries:

- Q1. What is the acceleration profile along a given route segment for a given trip?*
- Q2. What is the difference between the vehicle's speed profile and the speed limit along a road segment?*
- Q3. How many times was the ABS enabled for a given trip?*
- Q4. What are the trips where the practiced speed exceeds a specified speed profile (e.g., the speed limit) by a certain value and what is the difference?*
- Q5. What is the ratio between speed and engine RPM for a given trip?*
- Q6. What is the average profile of acceleration for all vehicles passing through a certain road section (e.g., curve)?*
- Q7. Calculate the maximal speed profile of all vehicles passing through the indicated road section.*
- Q8. Find the average speed profile actually practiced (85th percentile of the passing vehicles) on a road before and after the installation of a speed camera.*
- Q9. What is the average profile of the fuel consumption on a road before and after the installation of a traffic calming device (e.g., a speed cushion)?*
- Q10. What is the minimum and maximum profile of fuel consumption on a road, and what is its difference with the profile of the studied driver?*

Modeling temporal sequences is feasible by using functions over time [6], but it is not useful for the above type of analysis. Indeed, the measures from the trips are collected at different times and comparing these profiles makes sense only if they were measured in the same place. What matters is not the time at which the measure was recorded, but rather where it took place on the road. The concept of measure profile (speed, acceleration) reflects the relationship between the measure and space. However, this notion of profile could not be defined in the model Gütting or any other model. It is therefore necessary to extend the existing model with new data types. Moreover, the above queries demand specific operations on the measure profiles. This kind of operations that was not necessary in the context of analyzing only the MO trajectories, are of major importance in this context.

² The concept of profile is introduced in section 3.2

3.2. Introduction of New Data Types

Like the algebraic model in [5] described above, our model includes a spatio-temporal type to model the trajectory of the moving object, and a temporal type to model the data generated by sensors. The temporal type is a function of time to base types (*real*, *int*). It expresses the variability of sensor measures from the temporal point of view.

However, the temporal view is not sufficient to model the *data from mobile sensors*, because the measures are often closely related to space. For completeness, the model should describe beside the evolution over time, the spatial evolution of the measures.

To this end, we extend the model of [5]. We introduce a new concept describing the spatial profile of measures. The idea is to describe the evolution of a measure in space. This concept is divided into two categories: *SVARIANT* to describe the profile in a two-dimensional space, and *GVARIANT* for the profile along the network.

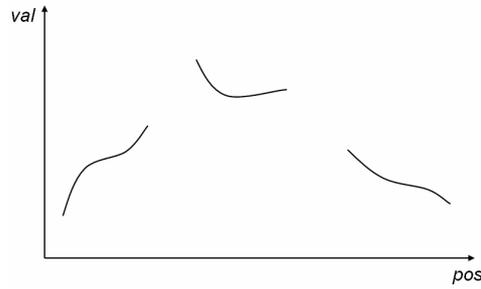


Fig. 1. Example of spatial profile of a real value.

We propose two new type constructors called *smoving* and *gmoving*:

Table 2. New data types.

$BASE \rightarrow SVARIANT$	<i>smoving</i>
$BASE \rightarrow GVARIANT$	<i>gmoving</i>

The definitions of these type constructors are given below using the notation of [6]:

Definition 1: Given α a *BASE* type having the carrier set A_α . Then the domain of definition for *smoving*(α) is defined as follows: $A_{smoving(\alpha)} = \{f \mid f : \bar{A}_{point} \rightarrow \bar{A}_\alpha \text{ is a partial function and } \wedge \Gamma(f) \text{ is finite } \}$, where $\bar{A}_\beta = A_\beta \setminus \{\perp\}$ and $\Gamma(f)$ denotes the set of maximal continuous components of the function f .

Definition 2: Given α a *BASE* type having the carrier set A_α . Then the domain of definition for *gmoving*(α) is defined as follows: $A_{gmoving(\alpha)} = \{f \mid f : \bar{A}_{gpoint} \rightarrow \bar{A}_\alpha \text{ is a partial function and } \wedge \Gamma(f) \text{ is finite } \}$, where $\bar{A}_\beta = A_\beta \setminus \{\perp\}$ and $\Gamma(f)$ denotes the set of maximal continuous components of the function f .

Since this article focuses on constrained movement, we only detail the second category of types in the sequel. These definitions state that a spatial profile of a measure is a partial function. Each value f in the domain of *gmoving*(α) is a function describing the evolution in the network (graph) space of a *BASE* value. The *gmoving* type constructor describes an infinite set of pairs (*position*, *value*), where the position is a *gpoint*. Figure 1 presents a spatial

profile of a real measure on a given road. The x-axis represents the relative position on the road that can vary between 0 and 1.

The condition " $\Gamma(f)$ is finite" means that f consists of only a finite number of continuous components. For example, the profile in Figure 1 has 3 continuous components. This condition is needed as a precondition to make the design implementable. It also ensures that projections of *gmoving* objects (e.g. on the spatial axis) have only a finite number of components.

The presented model is an *abstract model*, which means that in general the domains or carrier sets of its data types are infinite sets. To be able to implement an abstract model, one must provide a corresponding *discreet model*, i.e., define finite representation for all the data types of the abstract model. This is done by the *sliced representation* introduced in [3]. Thus, a time dependent or spatial dependent value is represented as a sequence of slices (see Figure 4) such that within each slice the evolution of the value can be described by some "simple" function (e.g., linear functions or quadratic polynomials). More details on the sliced representation are given in section 4 of the paper.

3.3. Introduction of New Operations

As for the type system definition, we use the operations in the algebra of Güting et al. [5] as a starting point. By introducing new types, we have to (i) extend the existing operations and (ii) add new specific operations for the target application type.

In order to extend the existing operations to the new type, we use a similar process with the *temporal lifting*, described in section 2. The temporal lifting permits generating from a non-temporal operation with the signature $\alpha_1 \times \alpha_2 \times \dots \times \alpha_n \rightarrow \beta$, the temporal equivalent one having the signature $\alpha'_1 \times \alpha'_2 \times \dots \times \alpha'_n \rightarrow moving(\beta)$, where $\alpha'_i \in \{\alpha_i, moving(\alpha_i)\}$. Each of the arguments can become temporal, which makes the result temporal as well. We adopt this principle to generate the equivalent space variant operations. We propose a *spatial lifting* for the non-gvariant non-temporal operations. The operation induced by the spatial lifting is available for a signature $\alpha'_1 \times \alpha'_2 \times \dots \times \alpha'_n \rightarrow gmoving(\beta)$, where $\alpha'_i \in \{\alpha_i, gmoving(\alpha_i)\}$.

We have also defined new operations that apply to *GVARIANT* and *TEMPORAL* types, which are necessary in this context. Table 3 presents a non-exhaustive list of the new operations, i.e., extended from the existing ones or newly introduced. We describe in this section some operations. Other operations are explained with the example queries in the next section. There are five classes of operations. The first two correspond to the extension of existing operations, while the last three are new types of operations. Moreover, the first four groups represent conventional operations, i.e., those who take as input one or more objects (values) in accordance with their signature and returns an object (a value). The last class includes aggregate operations, i.e., that return a single result based on a group of objects (similar to aggregates in the relational model).

The first class of operations comprises the projection in the network or value (range) domains. Thus, **trajectory** returns to the network path of a trip. The operation **rangevalues** performs the projection in the range and returns one or several intervals of base values. Operations **val** and **pos** return respectively the value or the network position for an *ingpoint* type, which is defined as a pair (*gpoint*, *value*). The second class of operations concerns the interaction with the domain (network space) and range (values). They make selections or clippings according to criteria on one of the axes of variation (network space or values). Thus, **present** is a predicate that checks if the input object is defined at a given position in the network. Finally, the predicate **passes** allows one to check whether the moving value ever assumed one of the values given as a second argument.

The third class of operations considers the basic algebraic operations ('+', '-', '!' and '/'), which we include in non-gvariant non-temporal collection of operations. Therefore, they become subject to temporal and spatial lifting. We use named functions, i.e., **sum**, **sub**, **mul** and **div**, as for all defined operations. These operations are useful for the analysis of sensor measures. For example, they can calculate the difference between the speed profiles of two moving objects on the common part of their trajectories, or return the difference between the practiced speed and the speed limit on a route. These operations take as input two functions of the same type (*GVARIANT* or *TEMPORAL*) and calculate a result function of which the definition domain is the intersection of the input objects' domains. For the division, the parts where the operation is not defined, are also eliminated from the domain of the result function.

Table 3. Examples of operations for new types.

Class	Operation	Signature
Projection to Domain/Range	trajectory	$gmoving(\alpha) \rightarrow gline$
	rangevalues	$gmoving(\alpha) \rightarrow range(\alpha)$
	pos	$ingpoint \rightarrow gpoint$
	val	$ingpoint \rightarrow \alpha$
Interaction with Domain/Range	atpos	$gmoving(\alpha) \times gpoint \rightarrow ingpoint$
	atgline	$gmoving(\alpha) \times gline \rightarrow gmoving(\alpha)$
	present	$gmoving(\alpha) \times gpoint \rightarrow bool$
		$gmoving(\alpha) \times gline \rightarrow bool$
	at	$gmoving(\alpha) \times \alpha \rightarrow gmoving(\alpha)$
		$gmoving(\alpha) \times range(\alpha) \rightarrow gmoving(\alpha)$
	atmin	$gmoving(\alpha) \rightarrow gmoving(\alpha)$
atmax	$gmoving(\alpha) \rightarrow gmoving(\alpha)$	
passes	$gmoving(\alpha) \times \beta \rightarrow bool$	
Basic Algebraic Operations	sum, sub, mul, div	$moving(\alpha) \times moving(\alpha) \rightarrow moving(\alpha)$ $gmoving(\alpha) \times gmoving(\alpha) \rightarrow gmoving(\alpha)$
Calculations	mean[avg], min, max	$moving(\alpha) \rightarrow real$
		$gmoving(\alpha) \rightarrow real$
	no_transitions	$moving(int) \rightarrow int$ $gmoving(int) \rightarrow int$
Aggregates	min_agg, max_agg, sum_agg, avg_agg	$\{moving(\alpha)\} \rightarrow moving(\alpha)$
		$\{gmoving(\alpha)\} \rightarrow gmoving(\alpha)$
	percentile	$\{moving(\alpha)\} \times real \rightarrow moving(\alpha)$ $\{gmoving(\alpha)\} \times real \rightarrow gmoving(\alpha)$
count_agg	$\{moving(\alpha)\} \rightarrow moving(int)$	
	$\{gmoving(\alpha)\} \rightarrow gmoving(int)$	

The fourth class of operation addresses the same categories of types, i.e., *GVARIANT* or *TEMPORAL*. The specified functions are: **mean**, **min**, **max** and **no_transitions**. Each of these operations takes as input a function of time or space and returns a value representing the aggregate of the input function. Their interest is to calculate an average or an extreme value for any measure, given a temporal or spatial interval.

The last class of operations concerns the aggregates. Aggregate operations return a single object result given a set of objects of the same type (see Figure 2). Unlike the previous class, these operations define aggregations of a group of objects. Some of these aggregates return an object of the same as the input type, e.g., the average (**avg_agg**), the minimum (**min_agg**) and

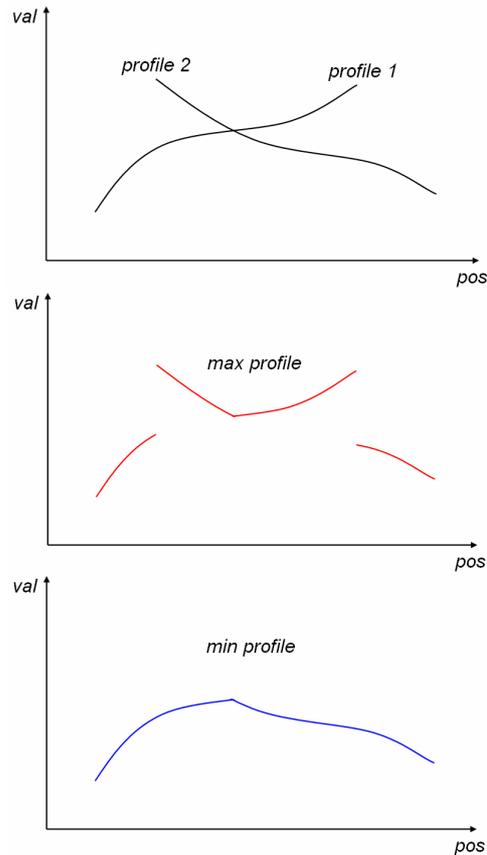


Fig. 2. Example of using `max_agg` (second graph) and `min_agg` (third graph) on two profiles (first graph).

maximum profile (**max_agg**). The aggregate **count_agg** returns the number of profiles in the definition domain in the form of a *moving(int)* or *gmoving(int)* object. Finally, the function **percentile** computes the profile below which is found a certain percentage of profiles in the input set. The definition domain of the result function for an aggregate operation is the union of domains of aggregated functions. The usefulness of aggregate operations is shown by the queries Q6 to Q10.

The proposed collection of operations is only a basis, however rich, of functionality. Other operations may be added to meet specific needs of some applications. Thanks to advances in the extensibility of the existing DBMS, these types and operations can be easily integrated into the DBMS. Thus, it becomes possible to use them through the standard SQL language. Then, the problem of optimizing queries must be addressed. This is exactly the plan that we followed to implement our data server for moving objects with sensors.

3.4. Query Examples

The great interest of using the extension capabilities of a DBMS is to easily integrate new types and operations in the SQL standard interface. The query examples in this section are based on a relational schema with one table that contains information on vehicle trips as follows:

```
vehicle_trip(mo_id: int, trip: moving(qpoint),
g_speed: gmoving(real), t_speed: moving(real), g_acceleration:
gmoving(real), t_acceleration: moving(real), g_RPM: gmoving(real), t_RPM:
moving(real), g_odometer: gmoving(real), t_odometer: moving(real), g_ABS:
```

```
gmoving(bool), t_ABS: moving(bool), g_breakSwitch: gmoving(real),  
t_breakSwitch: moving(real))
```

In addition to the spatio-temporal trajectory, i.e., the "trip", the table contains sensor data reporting the speed, acceleration, RPM, odometer, ABS and brake pedal state. These data are modeled by functions of space (prefixed with *g_*) and of time (prefixed with *t_*). The parameters are prefixed with the symbol "&" and could be either given by the user at runtime, or existing from previous calculations.

Q1. What is the acceleration profile along a given route segment for a given trip?

```
SELECT atgline(g_acceleration, &aGline)  
FROM vehicle_trip  
WHERE mo_id = &anID
```

The operation **atgline** returns the longitudinal acceleration profile restricted to the sub-space specified by the geometry *aGline* given as parameter.

Q2. What is the difference between the vehicle's speed profile and the speed limit along a road segment?

```
SELECT sub(g_speed, &legalSpeed)  
FROM vehicle_trip  
WHERE inside(  
    trajectory(&legalSpeed), trajectory(g_speed))=1
```

The difference between two functions describing measure profiles is calculated using the operation **sub**. An indexed predicate as **inside** could accelerate the response time of the query. This operation has two *gline* parameters and check if the first is included in the second. To obtain the projection in space of a measure profile, we use the operation **trajectory**.

Q3. How many times was the ABS enabled for a given trip?

```
SELECT no_transitions(g_ABS)/2  
FROM vehicle_trip  
WHERE mo_id = &anID
```

This query simply illustrates the use of **no_transitions**, which is applicable to discrete *BASE* types (e.g., *bool*, *int*) and returns the number of transitions for a given discrete function.

Q4. What are the trips where the practiced speed exceeds a specified speed profile (e.g., the speed limit) by a certain value and what is the difference?

```
SELECT mo_id, sub(g_speed, &legalSpeed)  
FROM vehicle_trip  
WHERE intersects(trajectory(&legalSpeed),  
    trajectory(g_speed)) = 1  
    AND max(sub(g_speed, &legalSpeed)) > &threshold
```

There are two new operations in this query. First, the predicate **intersects** is similar to **inside**, the only difference being that it only searches for an intersection between the two *gline* parameters and not for inclusion. Second, the operation **max** is an aggregate of a function. We use it to verify if the maximal value assumed by the function given as parameter is above a certain threshold value. As in the previous query, the parameter for **max** is represented by the difference between the practiced and legal speed profiles.

Q5. What is the ratio between speed and engine RPM for a given trip?

```
SELECT div(t_speed, t_RPM)  
FROM vehicle_trip  
WHERE mo_id = &anID
```

This query shows the usefulness of basic algebraic operations for comparing temporal profiles of the same moving object. The profile obtained by dividing the vehicle speed to the engine RPM can be used to detect the behavior regarding the gear shifting of a driver.

Q6. *What is the average profile of acceleration for all vehicles passing through a certain road section (e.g., curve)?*

```
SELECT avg_agg(g_acceleration)
FROM vehicle_trip
WHERE inside(trajjectory(&aCurve), trajjectory(trip))=1
```

We determine with this query the average acceleration profile of all vehicles passing through the indicated route section. The function **avg_agg** generates a new *gmoving(real)* object from the set of objects of the same type, passed as a parameter, i.e., all tuples of the table that match the predicate in the WHERE clause.

Q7. *Calculate the maximal speed profile of all vehicles passing through the indicated road section.*

```
SELECT max_agg(atgline(g_speed, &aRoad))
FROM vehicle_trip
WHERE intersects(trajjectory(g_speed), &aRoad) = 1
```

First we find all the trips that intersect the given road. For these trips, we select by the function **atgline** the speed profile that corresponds to the road. Then we aggregate the resulted profiles in order to obtain the maximal profile, by using the aggregate **max_agg**.

Q8. *Find the average speed profile actually practiced (85th percentile of the passing vehicles) on a road before and after the installation of a speed camera.*

```
SELECT percentile(atgline(g_speed, &aRoad), 85)
FROM vehicle_trip
WHERE intersects(trajjectory(g_speed), &aRoad) = 1
      AND inst(initial(trip)) < &instalationDate
```

The query finds the average speed profile (85th percentile) before installing a speed camera. A similar query should be posed to find the same profile after the installation of the camera. As for the query Q7, we filter the trips by retaining only those that intersect the given road, and that begin before the installation date of the camera. To do this we use the combination of functions **inst** and **initial** that return the start date of a trip. Finally, we apply the **percentile** function on all selected profiles. The second parameter of this function is the n^{th} percentile.

As we can see from this section, the new type and operations are needed to express these queries.

4. Implementation

In this section we address some implementation issues of the presented model and language that we currently implement as an extension of a DBMS. The objective is to offer a general view regarding some implementation aspects, rather than a thorough, detailed presentation. Thus, Section 4.1 presents the database system architecture. Section 4.2 details the sliced representation of the abstract data types. Sections 4.3 and 4.4 deal with optimization of the aggregate operation and predicate operations.

4.1. Database System Architecture

Currently, the support for spatio-temporal data in the existing DBMS is limited. However, most DBMS today offer possibilities for extensions to meet the needs of certain application domains. Rather than developing a prototype from scratch, we chose to implement the proposed model under such an existing system, i.e., the Oracle DBMS. Thus, all types are implemented as new object types in Oracle 11g. The operations are implemented in Java (Oracle DBMS integrates a Java Virtual Machine) and stored as a package in the database. These operations can be used in SQL queries along with the existing operations in the DBMS.

Finally, some filtering operations, i.e., operations used to identify the MOs that verify a certain spatial, temporal or on-value predicate, are indexed in order to accelerate the query response time and to provide a system scalable with the dataset size (see Section 4.4). We integrate the indexes by using the data cartridges in Oracle. The general architecture of the system is given in Figure 3. Notice that other DBMS systems that allow DBMS extensions can be used. For example, we currently test the system portability to PostgreSQL.

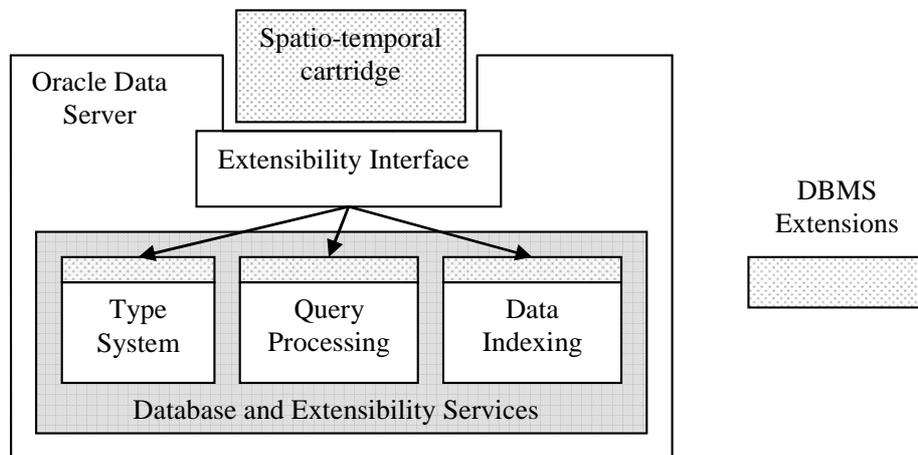


Fig. 3. Database system architecture

4.2. Data Type Representation

The model in [5], [6] that we extended in section 3.2 is an abstract model. A finite representation of this abstract model is needed in order to be able to implement it. For all *moving* types, the so-called *sliced representation* has been proposed in [3]. A *moving* object in the abstract model is a temporal partial function. The sliced representation represents the moving object as a set of so-called *temporal units* or *slice*. Figure 4 offers a simple example of a temporal profile that is composed of four *units*. A temporal unit for a moving data type α is a time interval where values taken by an instance of α can be described by a “simple” function. The “simple” functions used for the representations are the linear function or quadratic polynomials. The motivation for this choice is a trade-off between richness of the representation and simplicity of the representation of the discrete type and of its operations. For example, a unit for a *moving(real)* object is represented as a tuple $(a, b, c, t1, t2)$, where a, b, c are the coefficients of a quadratic polynomial and $(t1, t2)$ is the unit time interval. The moving value at a time instant t inside the unit time interval is computed as $a \times t^2 + b \times t + c$. More complex function for unit representation can be imagined but are not considered in this paper. Also, for the sake of simplicity we ignore that the unit intervals are left-closed and/or right-closed.

For all *moving* types that we introduced in Section 3.2, we adopt the sliced representation as proposed in [3]. This is straightforward as the sole difference is to replace the unit time interval, which is the support for temporal profiles, with a unit spatial interval, which is the support for spatial profiles. A spatial interval given a network space has the following elements: $(rid, pos1, pos2)$, where *rid* is a road identifier and *pos1*, *pos2* are relative positions on the road. For example, *moving(real)* object will contain a set of units with the following attributes: $(a, b, c, rid, pos1, pos2)$. To calculate a value for a given position, we first locate the corresponding unit, i.e., where the spatial interval includes the position, then calculate the value as $a \times pos^2 + b \times pos + c$.

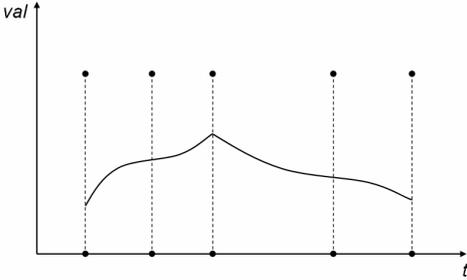


Fig. 4. Example of sliced representation of a temporal profile.

4.3. Handling Aggregation Functions

Unlike the functions on one or two profiles, aggregate functions operate on a set containing a (possibly) large number of profiles. This can lead to a fragmentation of the result profile in a large number of small units and a degradation of performance. Consider the example in Figure 5. The first graph shows two profiles with their decomposition into units and the second one represents the maximum aggregate of these profiles. Notice that the units of the two profiles do not have the same spatial distribution. The space intervals of the units of the two profiles overlap partially. This is common because the unit slicing is unique to each

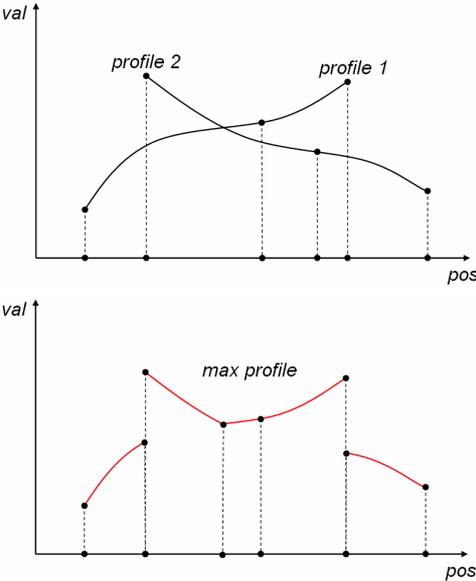


Fig. 5. Example of fragmentation after using the **max_agg** (second graph) on two profiles (first graph).

profile and depends on the variability of the observed measure at the observation time. Therefore, the result of an operation on a set of profiles is another profile that contains more units than the initial profiles. This process of fragmentation of the result is not disturbing when the calculation is done only on two profiles. However, in the case of aggregation it can significantly slow down the computation time.

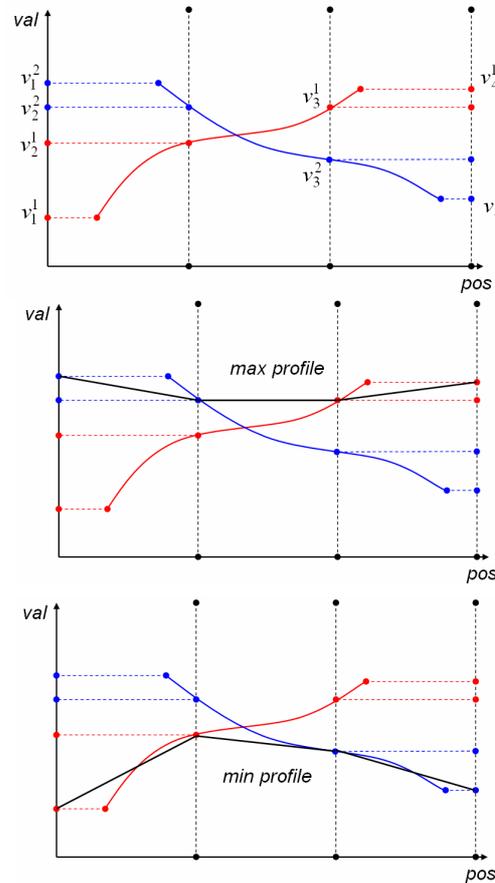


Fig. 6. Example of calculating the `max_agg` (second graph) and `min_agg` (third graph) on two profiles (first graph) using a regular slicing.

To accelerate the aggregate operations we propose a regular temporal or spatial slicing of profiles, independent of the initial slicing. This method offers a compromise between efficiency and quality of results. Thus, for aggregates on *moving* types for example, we uniformly divide the space, beginning with the start point of each road in intervals of a given length, e.g., 10 meters. Smaller intervals will produce higher quality results but at a cost of a slower performance, and vice versa.

Figure 6 presents an example of using uniform slicing for computing aggregates on two spatial profiles. The first graph shows the profile decomposition by regular intervals (represented by the vertical dotted lines). For each interval, we compute or extrapolated first the values on the end limits of the interval. Thus, for the first profile (in red) we find the values v_1^1 and v_2^1 the first interval, v_2^1 and v_3^1 for the second interval, and v_3^1 and v_4^1 for the third interval. From these values computed for all profiles, we apply the corresponding scalar aggregate function (e.g. the aggregate max for `max_agg`) in order to generate the values of the resulting profile on the same limits of the intervals.

Overall, using this approach to implement the aggregate functions produces approximate results, but it offers in return a good optimization of this costly type of operation. The analysis of the result quality depending on the granularity of slicing is left for future work.

4.4. Operators

In the field of spatio-temporal databases, the indexing techniques that permit processing efficiently the spatial, temporal and on-value queries are complementary to modeling the moving objects. We do not discuss the indexing methods in the spatio-temporal context as this is out of the scope of the paper. Instead, we present in this section the mechanism by which the indexes are linked to the implemented algebra, i.e., object types and operations. This mechanism is based on *operators*.

Operators are a subset of the algebra operations, mostly predicates such as **present** or **passes**, that benefit of an index based evaluation in addition to the basic function implementation. The functional implementation is used when the operator is invoked in the select list of a SELECT command or in the ORDER BY and GROUP BY clauses. However, when the operator appears in the condition of a WHERE clause, the DBMS optimizer chooses between the index implementation and the functional implementation, taking into account the selectivity and cost while generating the query execution plan.

The operators that we implement are spatial, temporal and on-value predicates, or predicates that combine two of the three possible dimensions (i.e., spatio-temporal, on-value spatial and on-value temporal). For example, to select only the profiles that spatially intersect a given network region, one will use the **present** operator. In the same way, the **passes** operator is used to select only the objects for which a certain measure assumes a given value (e.g., acceleration is above 10m/s^2). Finally, the two-dimensional predicates verify that the conditions in each dimension are simultaneously verified. For example, a spatio-temporal operator ensures that the trajectory of a MO intersects (or is included) a (in) spatial network region at a given time interval. Similar reasoning can be applied for the rest of the operators.

5. Conclusions and Future Work

The use of sensors embedded in vehicles leads to new applications, which give rise to new research problems. In this paper, we address the problem of modeling and querying. We have stressed the limitations of existing work on moving objects. The new type of indicated applications concerns moving objects equipped with sensors that produce additional data streams, which are strongly related to the trajectory of the object. A DBMS capable of managing in a unified manner the data concerning the moving object and the corresponding data flows is very useful for these applications.

The contribution of this paper is to propose a model for such a DBMS by extending an existing framework for moving objects. We first analyzed the limitations of modeling sensor data. Indeed, existing models can represent the data flows from a temporal point of view. We have shown that these measures are equally dependent of object's position and a representation relative to space is needed. Therefore, we have extended the existing type system with functions that describe the evolution of measures in space. We have also proposed a collection of operations in view of the enhanced system. We introduced the concept of *spatial lifting* inspired by the idea of the existing *temporal lifting*. We have redefined all the temporal operations and changed the semantics of some of them for the new types. Finally, we proposed new operations, and aggregates on the new types. An illustration of use of the DBMS is given by query examples involving the new defined types and

operations. The current prototype includes a partial implementation of the algebra as a data cartridge in Oracle DBMS.

We presented in [18] a demonstration for a real application related to naturalistic driving in conjunction with intelligent transportation systems and the results are promising for a scenario on a larger scale.

The presented application type usually generates very large datasets. As future work, we intend to study proper indexing techniques for the new types. Although this is similar to the query optimization problem in moving object databases, sensor data types and distribution may lead to specific optimizations in our system. We also investigate the problem of mining such databases [9].

Acknowledgments. This work was supported by grants from Région Ile-de-France. We would like to thank LIVIC laboratory (affiliated to INRETS and LCPC) for their collaboration in the application scenario.

References

1. Burns, P.C.: International Harmonized Research Activities – Intelligent Transport Systems (IHRA-ITS) working group report. 19th International Technical Conference on the Enhanced Safety of Vehicle (ESV), 2005
2. Ehrlich, J., Marchi, M., Jarri, P., Salesse, L., Guichon, D., Dominois, D., Leverger, C.: LAVIA, the French ISA project: Main issues and first results on technical tests, 10th World Congress & Exhibition on ITS, November 2003
3. Forlizzi, L., Güting, R.H., Nardelli, E., and Schneider, M.: A Data Model and Data Structures for Moving Objects Databases. ACM SIGMOD: 319-330, 2000
4. Grumbach, S., Rigaux, P., Segoufin, L.: Spatio-Temporal Data Handling with Constraints, *GeoInformatica*, 5(1), 2001
5. Güting R.H., Almeida, V.T., and Ding Z.: Modeling and Querying Moving Objects in Networks. *VLDB Journal* 15(2): 165-190, 2006
6. Güting, R.H., Böhlen, M.H., Erwig, M., Jensen, C.S., Lorentzos, N.A., Schneider, M., and Vazirgiannis, M.: A Foundation for Representing and Querying Moving Objects. *ACM Transactions on Database Systems*, 25(1): 1-42, 2000
7. Güting R.H., Schneider M.: *Moving Objects Databases*, Morgan Kaufmann, 2005
8. ISO 19107:2003, *Geographic Information – Spatial Schema*, WG 2
9. Kharrat, A., Sandu Popa, I., Zeitouni, K., Faiz, S.: Clustering algorithm for network constraint trajectories. The 13th International Symposium on Spatial Data Handling, SDH 2008
10. Koubarakis, M., Pernici, B., Schek, H.J., Scholl, M., Theodoulidis, B., Tryfona, N., Sellis, T., Frank, A.U., Grumbach, S., Güting, R.H., Jensen, C.S., Lorentzos, N., Manolopoulos, Y., and Nardelli, E. (Eds.): *Spatio-Temporal Databases: The CHOROCHRONOS Approach*. Springer-Verlag, *Lecture Notes in Computer Science* 2520, 2003
11. NHTSA (2006). The 100-Car Naturalistic Driving Study, Phase II – Results of the 100-Car Field Experiment. Report No. DOT HS 810 593
12. Oracle Database Data Cartridge Developer's Guide, 11g Release 1, 2007
13. Pelekis, N.: STAU: A Spatio-Temporal Extension to ORACLE DBMS, PhD Thesis UMIST, Department of Computation, 2002
14. Pelekis, N., Frenzos, E., Gitrakos, N., Theodoridis, Y.: HERMES: Aggregative LBS via a Trajectory DB Engine, *SIGMOD 2008*: 1255-1258
15. Pelekis, N., Theodoridis, Y.: Boosting Location-Based Services with a Moving Object Database Engine, *Proceedings of MobiDE*, 2006
16. Pelekis, N., Theodoulidis, B., Kopanakis, I., Theodoridis, Y.: Literature Review of Spatio-Temporal Database Models, *The Knowledge Engineering Review Journal*, 19(3), 235-274, June 2005

17. Pfoser, D., and Jensen, C.S.: Indexing of Network-constrained Moving Objects. ACM-GIS, 25-32, 2003
18. Sandu Popa, I., Zeitouni, K., Dupin, F., Saint-Pierre, G., Glaser, S.: Using in-Vehicle Sensor Data for Naturalistic Driving Analysis, 15th World Congress on Intelligent Transportation Systems (ITS'2008), 2008
19. Savary, L., Wan, T., Zeitouni, K.: Spatio-Temporal Data Warehouse Design for Activity Pattern Analysis, Int. DEXA Workshop on Geographic Information Management, 2004
20. Sistla, P., Wolfson, O., Chamberlain, S., Dao, S.: Modeling and Querying Moving Objects, Proc. 13th Int'l Conf. on Data Engineering (ICDE13), Birmingham, UK, 1997
21. Vazirgiannis, M., and Wolfson, O.: A Spatiotemporal Query Language for Moving Objects. Proc. of SSTD: 20-35, 2001
22. Wolfson, O.: Moving Objects Information Management: The Database Challenge. NGITS: 75-89, 2002
23. Yeh, T.S., and B. de Cambray: Modeling Highly Variable Spatio-Temporal Data. 6th Australasian Database Conference (ADC'1995), 221-230, 1995